

Deciding Hedged Bisimilarity

Alessio Mansutti and Marino Miculan

Laboratory of Models and Applications of Distributed Systems
 Department of Mathematics, Computer Science and Physics, University of Udine
alessio.mansutti@gmail.com marino.miculan@uniud.it

Abstract. The spi-calculus is a formal model for the design and analysis of cryptographic protocols: many security properties, such as authentication and strong confidentiality, can be reduced to the verification of behavioural equivalences between spi processes. In this paper we provide an algorithm for deciding *hedged bisimilarity* on finite processes, which is equivalent to barbed equivalence (and coarser than framed bisimilarity). This algorithm works with any term equivalence satisfying a simple set of conditions, thus encompassing many different encryption schemata.

Keywords: security, cryptographic protocols, spi-calculus, bisimilarity.

1 Introduction

The spi calculus, introduced by Abadi and Gordon in [2], is a process calculus designed for the description and formal verification of cryptographic protocols. Many security properties, such as authentication and strong confidentiality, can be reduced to the verification of *may-testing equivalences* between spi processes. Since may-testing equivalences are difficult to check in practice, other behavioural equivalences have been put forward for this calculus. In [1] Abadi and Gordon defined *framed bisimilarity*, a bisimulation-style equivalence which is a sound approximation of may-testing equivalence. Later, other context-sensitive equivalences have been proposed; in particular, Borgström and Nestmann defined *hedged bisimilarity* [3], which is shown to be equivalent to *barbed equivalence* and strictly coarser than framed bisimilarity. We refer to [3] for a detailed comparison of these equivalences, which we summarize in Figure 1.

Hüttel [4] proved that framed bisimilarity is decidable on finite processes. In this paper, we extend this result, proving that also hedged bisimilarity (and hence barbed equivalence) is decidable on finite processes. Moreover, we do not choose a specific congruence over terms; rather, the algorithm works with any congruence relation, as long as some mild conditions are satisfied. Therefore, our algorithm can be readily applied to different encryption/decryption schemata just by changing the congruence rules. These conditions are introduced in Section 2, where we recall also the syntax and *late* operational semantics of spi-calculus. In Section 3 we define the notion of *hedged bisimilarity* using this late semantics, and in Section 4 we show that it is decidable on *finite* spi-calculus processes (i.e. processes without replication); an algorithm in pseudo-code is provided. Some concluding remarks and directions for future work are in Section 5.

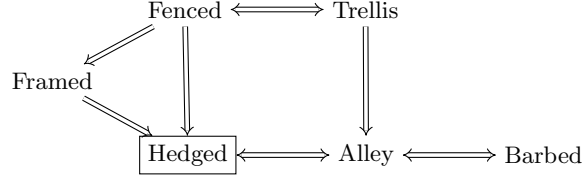


Fig. 1. Behavioural equivalences for the spi calculus [3].

2 The spi calculus

The *spi-calculus* extends the π -calculus with terms and primitives for encryption and decryption. In this section we define the variant we consider in this paper.

2.1 Syntax

Terms We first define the set of terms that can be used by processes, following [3].

Definition 1 (Terms). Let \mathcal{N} be a countable set of names ranged over by $a, b, c, n \dots$, and V a countable set of variable symbols, ranged over by $x, y, z \dots$. The set of *spi-calculus* terms is given by the grammar

$$\begin{aligned}
 A &::= a \mid x \\
 t &::= A \mid (t_1, t_2) \mid \pi_1(t) \mid \pi_2(t) \mid \{t_1\}_{t_2} \mid D_{t_2}(t_1) \\
 \phi &::= \mathbf{true} \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid [t_1 = t_2]
 \end{aligned}$$

Intuitively, $\{t_1\}_{t_2}$ denotes the term t_1 encrypted using key t_2 , and (t_1, t_2) denotes the pair whose components are terms t_1 and t_2 . Correspondingly, we have the destructor $D_{t_2}(t_1)$, which decrypts t_1 using key t_2 , and the two projections $\pi_1(t)$, $\pi_2(t)$. We use **false** as a shorthand for $\neg\mathbf{true}$.

The set of (free) variables of a term t is denoted by $\text{fv}(t)$; notice that there are no binding operators in terms. As usual, a term t is said to be *ground* if $\text{fv}(t) = \emptyset$, i.e. without (free) variables. It is said to be a *message* if it is ground and without occurrences of $\pi_1(\cdot)$, $\pi_2(\cdot)$ and $D(\cdot)$ operators. We will denote with \mathcal{M} the set of all messages, ranged over by M, N .

Unlike [2, 4], our terms are typed. Types are defined by the following syntax:

$$\tau ::= \mathbf{N} \mid \mathbf{B} \mid \tau_1 \times \tau_2 \mid \mathbf{C}(\tau)$$

where \mathbf{N}, \mathbf{B} are the base types of names and booleans respectively, and $\mathbf{C}(\tau)$ is the type of encrypted terms of type τ . Formally, the typing judgment $t : \tau$ over ground terms is defined by the following rules.

$$\begin{array}{c}
 \frac{a \in \mathcal{N}}{a : \mathbf{N}} \quad \frac{}{\mathbf{true} : \mathbf{B}} \quad \frac{\phi : \mathbf{B}}{\neg\phi : \mathbf{B}} \quad \frac{\phi_1 : \mathbf{B} \quad \phi_2 : \mathbf{B}}{\phi_1 \wedge \phi_2 : \mathbf{B}} \quad \frac{}{[t_1 = t_2] : \mathbf{B}} \\
 \frac{t_1 : \tau_1 \quad t_2 : \tau_2}{(t_1, t_2) : \tau_1 \times \tau_2} \quad \frac{t : \tau_1 \times \tau_2}{\pi_i(t) : \tau_i} \quad i = 1, 2 \quad \frac{t_1 : \tau \quad t_2 : \mathbf{N}}{\{t_1\}_{t_2} : \mathbf{C}(\tau)} \quad \frac{t_1 : \mathbf{C}(\tau) \quad t_2 : \mathbf{N}}{D_{t_2}(t_1) : \tau}
 \end{array}$$

Congruence over terms Terms are taken up-to some structural congruence \equiv , whose aim is to express the evaluation internal to processes, in particular the execution of encryption/decryption algorithms. Differently from [2–4], we aim to account for different type of encryption algorithms that can be expressed by choosing the structural equivalence \equiv . To this end, we provide a general definition of “coherent” congruence:

Definition 2 (Coherent Congruence). *A congruence relation \equiv over terms is coherent if the following hold:*

1. (Type preservation) $\forall t_1 : \tau_1, t_2 : \tau_2 \forall j, k \in \mathcal{N} : \{t_1\}_k \equiv \{t_2\}_j \Rightarrow \tau_1 = \tau_2$
2. (Equivariance) $\forall t_1, t_2 : \mathbb{C}(\tau) \forall a \in \mathcal{N} \forall b \in \mathcal{N} \setminus (n(t_1) \cup n(t_2)) : t_1 \equiv t_2 \iff t_1\{b/a\} \equiv t_2\{b/a\}$
3. (Deterministic decryption) $\forall t_1, t_2 : \mathbb{C}(\tau) : t_1 \equiv t_2 \Rightarrow \text{ds}(t_1) = \text{ds}(t_2)$

where $n(t)$ is the set of all names occurring in t , $t\{s/a\}$ is the syntactical substitution replacing all occurrences of a in t with the term s , and $\text{ds}(\cdot)$ is defined inductively by the clauses

$$\text{ds}(n \in \mathcal{N}) = n \quad \text{ds}((t_1, t_2)) = (\text{ds}(t_1), \text{ds}(t_2)) \quad \text{ds}(\{t_1\}_{t_2}) = \text{ds}(t_1)$$

The first condition says that two encrypted terms can be considered equal only if they encrypt messages with the same type. The second condition imposes the absence of special names (and keys): if a property holds for a name, then it must hold for every fresh name. The third condition says that congruence must be consistent with decryption: the decryption of a message M is thus guaranteed to be deterministic w.r.t. all messages in the same equivalence class of M .

It is easy to check that the equivalence used in [2–4] respects these conditions. Other encryption algorithms (and other abstract data types) can be considered by adapting the congruence relation, as long as it remains coherent; for example, we can analyze encryption protocols with commutative ciphers (like RSA) by adding the axiom $\forall M : \tau \forall k, j \in \mathcal{N} : \{\{M\}_k\}_j \equiv \{\{M\}_j\}_k$.

Processes We can now define the processes of the spi-calculus.

Definition 3 (Processes). *The spi-calculus processes are defined as follows:*

$$P ::= 0 \mid A(x).P \mid \overline{A}\langle t \rangle.P \mid P_1 \mid P_2 \mid P_1 + P_2 \mid (\nu a)P \mid !P \mid \phi.P \mid \text{let } x = t \text{ in } P$$

where t and ϕ are respectively a well-typed term and a boolean formula, x is a variable, a is a name and A can be both a name or a variable.

The syntax above are the usual ones from π -calculus, with these differences:

- input/output operations exchange terms, not only names and variables;
- $\phi.P$ is the *guard operator*, that behaves as P if the boolean formula ϕ holds;
- $\text{let } x = t \text{ in } P$ is the *let operator* that computes the value of t , assigns it to the variable x and then executes P .

Without loss of generality, we can assume that destructors $(\pi_1(\cdot), \pi_2(\cdot))$ and $D(\cdot)$ do not occur in boolean predicates $[t_1 = t_2]$, nor in the argument of output operations $\bar{A}(t).P$, since these cases can be simulated using the *let*. For instance, $\bar{a}(\pi_1(t)).P$ is equivalent to *let* $x = \pi_1(t)$ *in* $\bar{a}(x).P$, for $x \notin \text{fv}(P)$.

Processes are taken up-to the usual structural congruence familiar from the π -calculus theory:

$$\begin{array}{ll}
P \equiv Q & \text{if } P \text{ and } Q \text{ are } \alpha\text{-equivalent} & !P \equiv P!P \\
P|Q \equiv Q|P & & (P|Q)|R \equiv P(Q|R) \\
P + Q \equiv Q + P & & (P + Q) + R \equiv P + (Q + R) \\
(\nu m)(P|Q) \equiv (\nu m)P|Q & \text{if } m \notin \text{fn}(Q) & (\nu m)(\nu n)P \equiv (\nu n)(\nu m)P \\
(\nu m)0 \equiv 0 & P|0 \equiv P & \frac{P \equiv Q}{P|R \equiv Q|R} \quad \frac{P \equiv Q}{P + R \equiv Q + R} \quad \frac{P \equiv Q}{(\nu m)P \equiv (\nu m)Q}
\end{array}$$

2.2 Semantics

To define the operational semantics of the spi-calculus we need to evaluate terms and boolean expressions. Evaluation is defined over *well-typed* terms, where each type denotes a set of values:

Definition 4 (Interpretation of types). *The interpretation of types $\llbracket \cdot \rrbracket : \text{Types} \rightarrow \text{Set}$ is defined recursively as follows:*

$$\begin{aligned}
\llbracket \mathbf{N} \rrbracket &= \mathcal{N} \\
\llbracket \mathbf{B} \rrbracket &= \{\mathbf{true}, \mathbf{false}\} \\
\llbracket \mathbf{C}(\tau) \rrbracket &= \{\{M\}_k \mid M : \mathbf{C}(\tau), k \in \mathcal{N}\} / \equiv \\
\llbracket \tau_1 \times \tau_2 \rrbracket &= \{(M_1, M_2) \mid M_1 \in \llbracket \tau_1 \rrbracket, M_2 \in \llbracket \tau_2 \rrbracket\} / \equiv
\end{aligned}$$

Definition 5 (Evaluation). *The evaluation for ground terms and boolean expressions is a partial function $\llbracket \cdot \rrbracket : M_\tau \rightarrow \llbracket \tau \rrbracket$ (implicitly parametric in the type τ) defined recursively as follows:*

$$\begin{aligned}
\llbracket a \rrbracket &= a \in \mathcal{N} & \llbracket (t_1, t_2) \rrbracket &= (\llbracket t_1 \rrbracket, \llbracket t_2 \rrbracket) \\
\llbracket \mathbf{true} \rrbracket &= \mathbf{true} & \llbracket \pi_1(t) \rrbracket &= v_1 \text{ if } \llbracket t \rrbracket = (v_1, v_2) \\
\llbracket \phi_1 \wedge \phi_2 \rrbracket &= \llbracket \phi_1 \rrbracket \wedge \llbracket \phi_2 \rrbracket & \llbracket \pi_2(t) \rrbracket &= v_2 \text{ if } \llbracket t \rrbracket = (v_1, v_2) \\
\llbracket \neg \phi \rrbracket &= \neg \llbracket \phi \rrbracket & \llbracket \{t_1\}_{t_2} \rrbracket &= \{\llbracket t_1 \rrbracket\}_{\llbracket t_2 \rrbracket} \\
\llbracket [t_1 = t_2] \rrbracket &= \mathbf{true} \text{ if } \llbracket t_1 \rrbracket \equiv \llbracket t_2 \rrbracket; \mathbf{false} \text{ otherwise} \\
\llbracket D_{t_2}(t_1) \rrbracket &= t \in \tau \text{ if } k = \llbracket t_2 \rrbracket \in \mathcal{N} \text{ and } \llbracket t_1 \rrbracket \equiv \{t\}_k \in \llbracket \mathbf{C}(\tau) \rrbracket
\end{aligned}$$

We will write $\llbracket t \rrbracket \downarrow$ if the evaluation of t produces a value, $\llbracket t \rrbracket \uparrow$ otherwise – i.e. when the evaluation of a decryption or a projection is unsuccessful.

Following [2, 4] (and differently from [3]) we define a *late input* style operational semantics. We first define the *reduction* relation, which describes how

processes unfold and execute internal computations in preparation for a reaction.

$$\frac{\llbracket \phi \rrbracket = \mathbf{true}}{\phi.P > P} \quad \frac{\llbracket t \rrbracket = v}{\text{let } x = t \text{ in } P > P\{v/x\}} \quad \frac{P > P'}{P \equiv P'}$$

where v is the value of t (up-to congruence), if defined.

The next step is to define *abstractions* F and *concretions* C :

$$F ::= (x)P \quad C ::= (\nu m_1 \dots m_n)\langle M \rangle Q \quad n \geq 0$$

where the variable x is bound in P and names m_1, \dots, m_n are bound in M, Q . As we will see in the semantic rules, an input $a(x).P$ becomes an abstraction after performing a transition labeled a ; this abstraction can be seen as a process waiting to receive a message on channel a . An output $\bar{a}\langle M \rangle.Q$ becomes a concretion $(\nu \mathbf{m})\langle M \rangle Q$, where \mathbf{m} are fresh names that can appear in M and P . This concretion can be seen as a process waiting to send a message on the channel a .

An abstraction $(x)P$ and a concretion $(\nu \mathbf{m})\langle M \rangle Q$ can interact via synchronization resulting in a process where the message M is received by $(x)P$. In order to define this interaction, we need to extend restriction and parallel composition operators to abstractions and concretions, as follows:

$$\begin{aligned} (\nu m)(x)P &\triangleq (x)(\nu m)P \\ (\nu n)(\nu \mathbf{m})\langle M \rangle P &\triangleq \begin{cases} (\nu n, \mathbf{m})\langle M \rangle P & \text{if } n \in \text{fn}(M) \\ (\nu \mathbf{m})\langle M \rangle (\nu n)P & \text{otherwise} \end{cases} \\ R|(x)P &\triangleq (x)(R|P) \quad \text{where } x \notin \text{fv}(R) \\ R|(\nu \mathbf{m})\langle M \rangle P &\triangleq (\nu \mathbf{m})\langle M \rangle (R|P) \quad \text{where } \{\mathbf{m}\} \cap \text{fn}(R) = \emptyset \end{aligned}$$

where the two last definitions can be always applied, by α -conversion.

Finally, the operational semantics of the spi-calculus is represented by a labelled transition relation $P \xrightarrow{\alpha} D$, where D ranges over processes, concretions and abstractions, and $\alpha \in \{a, \bar{a} \mid a \in \mathcal{N}\} \cup \{\tau\}$ is the label. As usual, the transition labelled with τ is also called silent transition or τ -transition. The relation is defined by the rules given in Figure 2. As usual, we also define

$$P \Longrightarrow Q \triangleq P \xrightarrow{\tau}^* Q \quad P \xRightarrow{\alpha} Q \triangleq P \Longrightarrow \xrightarrow{\alpha} Q$$

3 Hedged bisimilarity

In this section we define the *hedged bisimilarity*, introduced in [3]. The basic idea is to mimic the *frame-theory* pairs of the *framed bisimilarity* defined in [1], but dropping the separate *frame* component and including corresponding names as part of the *theory*. The resulting theory is then called a *hedge*.

Definition 6 (Hedge). *Let \mathcal{M} be a set of messages. A hedge is a finite subset of \mathcal{M}^2 . We denote by \mathcal{H} the set of all hedges.*

A hedge h is consistent if and only if it is pair-free (i.e. all messages in h are not pairs) and whenever $(M, N) \in h$ we have that:

$$\begin{array}{l}
\text{(input)} \frac{a \in \mathcal{N}}{a(x)P \xrightarrow{a} (x)P} \quad \text{(output)} \frac{\llbracket t \rrbracket = M \quad a \in \mathcal{N}}{\bar{a}(t)P \xrightarrow{\bar{a}} (\nu)\langle M \rangle P} \\
\text{(interaction)} \frac{P \xrightarrow{n} (x)P' \quad Q \xrightarrow{\pi} (\nu \mathbf{m})\langle M \rangle Q' \quad \{\mathbf{m}\} \cap \text{fn}(P') = \emptyset}{P|Q \xrightarrow{\tau} (\nu \mathbf{m})(P'\{M/x\}|Q')} \\
\text{(restriction)} \frac{P \xrightarrow{\alpha} D \quad \alpha \notin \{m, \bar{m}\}}{(\nu m)P \xrightarrow{\alpha} (\nu m)D} \quad \text{(parallel)} \frac{P \xrightarrow{\alpha} P'}{P|Q \xrightarrow{\alpha} P'|Q} \\
\text{(sum)} \frac{P \xrightarrow{\alpha} P'}{P+Q \xrightarrow{\alpha} P'} \quad \text{(equivalence)} \frac{P \equiv Q \quad Q \xrightarrow{\alpha} Q' \quad Q' \equiv P'}{P \xrightarrow{\alpha} P'}
\end{array}$$

Fig. 2. Late operational semantics of the spi-calculus.

- $M \in \mathcal{N} \iff N \in \mathcal{N}$;
- for all $(M', N') \in h$, if $M \equiv M'$ or $N \equiv N'$ then $M = M'$ and $N = N'$;
- if $M \equiv \{M'\}_k$ e $N \equiv \{N'\}_j$ then $k \notin \pi_1(h)$ and $j \notin \pi_2(h)$.

The first condition requires a consistent hedge to match names with names. The second one requires that $\pi_1(h)$ and $\pi_2(h)$ are in bijection. The last one requires that all encrypted messages cannot be decrypted using keys in h : encrypted messages in a consistent hedge are not reducible.

Alongside (consistent) hedges, we define *synthesis*, *analysis*, *irreducibles*. The *synthesis* of a hedge h is the set of message pairs that can be built from h .

Definition 7 (Synthesis). *The synthesis $S(h)$ of a hedge h is defined as the least subset of \mathcal{M}^2 that satisfies:*

- $h \subseteq S(h)$;
- if $(M, N) \in S(h)$, $(k, j) \in S(h)$ and $k, j \in \mathcal{N}$ then $(\{M\}_k, \{N\}_j) \in S(h)$;
- if $(M_1, N_1) \in S(h)$ and $(M_2, N_2) \in S(h)$ then $((M_1, M_2), (N_1, N_2)) \in S(h)$.

We write $h \vdash M \leftrightarrow N$ for $(M, N) \in S(h)$, and in this case we say that M and N are homologous w.r.t. h .

The *analysis* of a hedge is the set of all message pairs obtained by “opening” the messages of h via decryption or projection. The *irreducibles* are those elements in the analysis of a hedge that cannot be reduced further. Formally:

Definition 8 (Analysis). *The analysis $A(h)$ of a hedge h is defined as the least set that satisfies:*

- $h \subseteq A(h)$;
- if $(\{M\}_k, \{N\}_j) \in A(h)$ and $(k, j) \in A(h)$ then $(M, N) \in A(h)$;
- if $((M_1, N_1), (M_2, N_2)) \in A(h)$ then $(M_1, M_2) \in A(h)$ and $(N_1, N_2) \in A(h)$.

Moreover, we define the irreducibles $I(h)$ of a hedge h as

$$\begin{aligned}
I(h) \triangleq & A(h) \setminus (\{(C, D) \in A(h) \mid C \equiv \{M\}_k, D \equiv \{N\}_j, (k, j) \in A(h)\} \\
& \cup \{((M_1, N_1), (M_2, N_2)) \in A(h)\})
\end{aligned}$$

It should be noted that all elements that can be reduced in the analysis can be derived from the irreducibles via synthesis, i.e. $S(I(h)) = S(A(h))$. Lastly, since every hedge is a finite set, its analysis and irreducibles are also finite.

Hedge simulations Let us recall that \mathcal{H} and \mathcal{P} are the set of all hedges and the set of all processes, respectively. A *hedged relation* \mathcal{R} is a subset of $\mathcal{H} \times \mathcal{P} \times \mathcal{P}$. We write $h \vdash PRQ$ when $(h, P, Q) \in \mathcal{R}$. Moreover, we say that \mathcal{R} is *consistent* if, for all $h \in \mathcal{H}$, $h \vdash PRQ$ implies that the hedge h is consistent.

Definition 9 (Hedged simulation). *A consistent hedged relation \mathcal{R} is a hedged simulation if, whenever $h \vdash PRQ$ we have that:*

- if $P \xrightarrow{\tau} P'$ then there exists Q' such that $Q \Longrightarrow Q'$ and $h \vdash P'\mathcal{R}Q'$;
- if $P \xrightarrow{\bar{a}} (\nu \mathbf{m})\langle M \rangle P'$ and $\{\mathbf{m}\} \cap (\text{fn}(P) \cup n(\pi_1(h))) = \emptyset$ then there exist $b \in \mathcal{N}$ and a concretion $(\nu \mathbf{n})\langle M \rangle Q'$ such that $h \vdash a \leftrightarrow b$, $\{\mathbf{n}\} \cap (\text{fn}(Q) \cup n(\pi_2(h))) = \emptyset$, $Q \xRightarrow{\bar{b}} (\nu \mathbf{n})\langle M \rangle Q'$ and $I(h \cup \{(M, N)\}) \vdash P'\mathcal{R}Q'$;
- if $P \xrightarrow{a} (x)P'$ then there exist $b \in \mathcal{N}$ and an abstraction $(y)Q'$ such that $h \vdash a \leftrightarrow b$, $Q \xRightarrow{b} (y)Q'$ and for all $B \subset \mathcal{N}$ finite such that $B \cap (\text{fn}(P) \cup \text{fn}(Q) \cup n(h)) = \emptyset$ and $h \cup id_B$ is consistent, for all pairs (M, N) of ground terms, if $h \cup id_B \vdash M \leftrightarrow N$ then $h \cup id_B \vdash P'\{M/x\}\mathcal{R}Q'\{N/y\}$.

The first condition requires that for each τ -transition from P there is a path of τ -transition from Q such that the two target processes are in the simulation \mathcal{R} . The second condition requires that for each output transition of P , labelled with \bar{a} , there is an output transition from Q labelled with \bar{b} (and possibly preceded by some silent transitions); moreover, a and b are homologous in h and the processes after the two output operations are paired in \mathcal{R} w.r.t. a consistent hedge that extends h by pairing the two messages M and N . The last condition requires that for each input transition of P with label a , there is an input transition from Q labelled with b (and possibly preceded by some silent transitions); moreover, a and b are homologous in h and for all finite set B of fresh names w.r.t. P , Q and h , the abstractions $(x)P'$ and $(x)Q'$ are paired in the simulation \mathcal{R} for each input messages (M, N) homologous by $h \cup id_B$.

Definition 10 (Hedged bisimulation and bisimilarity). *A hedged simulation \mathcal{R} is a hedged bisimulation if $\mathcal{R}^{-1} = \{(h^{-1}, Q, P) \mid h \vdash PRQ\}$ is also a hedged simulation (where $h^{-1} = \{(N, M) \mid (M, N) \in h\}$).*

Hedged bisimilarity, written \sim , is the greatest hedged bisimulation, i.e. the union of all hedged bisimulations.

Remarkably, hedged bisimilarity coincides with barbed bisimilarity [3].

4 Decidability of hedged bisimulation for finite processes

Definition 9 does not provide us with a means for checking bisimilarity. In this Section we address this issue, following, when possible, the approach in [4].

Clearly, bisimilarity is undecidable for general, infinite processes; hence, we focus on *finite* processes, i.e. without replication. Even on finite processes decidability of hedged bisimilarity is not obvious, since the third condition in Definition 9 requires to check the equivalence of two abstractions for an infinite number of messages w.r.t. any finite set of fresh names. In this section we prove that there is a finite bound on the number of these names and messages. If this bound exists, then the hedged bisimilarity is trivially decidable.

The idea behind our result, as in [4], is the following: if $(x)P$ is finite, then it can inspect a message (using *let* and *guard* operators) up to a certain depth k . If a message M with more than k nested constructors is received by $(x)P$, then it can only be partially analysed by P . Hence, all messages M' equivalent to M up to depth k will not cause any difference in the execution of $(x)P$, apart from output messages. Indeed, $P\{M/x\}$ and $P\{M'/x\}$ can output different messages (i.e. different parts of M and M' respectively), but we notice that:

- the two outputs are derived from M and M' by applying the same operations;
- only messages obtained through decryption are interesting, since they can update the hedge h yielding a richer theory.

We now proceed to formalize this idea.

Definition 11 (Maximal constructor depth). *The maximal constructor depth $\text{mcd}(M)$ of a message M is defined inductively by the clauses*

$$\begin{aligned} \text{mcd}(n \in \mathcal{N}) &= 0 & \text{mcd}((M_1, M_2)) &= \max(\text{mcd}(M_1), \text{mcd}(M_2)) + 1 \\ \text{mcd}(x \in \mathcal{V}) &= 0 & \text{mcd}(\{V\}_K) &= \text{mcd}(V) + 1 \end{aligned}$$

and then extended to boolean formulas as follows:

$$\begin{aligned} \text{mcd}(\mathbf{true}) &= 0 & \text{mcd}(\phi_1 \wedge \phi_2) &= \max(\text{mcd}(\phi_1), \text{mcd}(\phi_2)) \\ \text{mcd}(\neg\phi) &= \text{mcd}(\phi) & \text{mcd}([M = N]) &= \max(\text{mcd}(M), \text{mcd}(N)) \end{aligned}$$

Definition 12 (k -homologous). *Given $h \in \mathcal{H}$ and $M, N \in \mathcal{M}$, we define*

$$h \vdash_k M \leftrightarrow N \stackrel{\Delta}{\iff} h \vdash M \leftrightarrow N \text{ and } k = \max(\text{mcd}(M), \text{mcd}(N))$$

Whenever $h \vdash_k M \leftrightarrow N$ we say that M and N are k -homologous in h .

The notion of *maximal constructor depth* is readily extended to hedges:

$$\text{mcd}(h) \triangleq \max\{k \mid \exists(M, N) \in h : h \vdash_k M \leftrightarrow N\}$$

Lemma 1. *Let $h \in \mathcal{H}$ and $M, N \in \mathcal{M}$ be such that $\max(\text{mcd}(M), \text{mcd}(N)) = k$. If there is a finite set of names $B \subset \mathcal{N}$ such that*

- $B \cap n(h) = \emptyset$;
- $h \cup id_B$ is consistent;
- $h \cup id_B \vdash_k M \leftrightarrow N$

then there exists $B' \subset \mathcal{N}$ with the same properties and such that $|B'| \leq 2^k$.

Proof. If $|B| \leq 2^k$ or $h \vdash M \leftrightarrow N$, then the thesis follows trivially and $B' = B$ or $B = \emptyset$, respectively. Otherwise, M and N are in the synthesis $S(h \cup id_B)$ and, at worst, all names in M and N are in B . Since $k = \max(\text{mcd}(M), \text{mcd}(N))$ and both *encrypt* and *pairing* are binary constructors, M and N can be represented as binary trees with height k . A binary tree of height k has at most 2^k leafs, hence B can be reduced to a set B' such that $|B'| \leq 2^k$ without losing any property in the hypothesis. \square

Lemma 1 leads us to the definition of *d-hedged bisimulation*: a hedged bisimulation up to a bound d on message depth and a bound 2^d on fresh names.

Definition 13 (d-hedged simulation). For any integer $d \geq 0$, a consistent hedged relation \mathcal{R} is a *d-hedged simulation* if whenever $h \vdash PRQ$ we have that:

- if $P \xrightarrow{\tau} P'$ then there exists Q' such that $Q \Longrightarrow Q'$ and $h \vdash P'\mathcal{R}Q'$;
- if $P \xrightarrow{\bar{a}} (\nu \mathbf{m})\langle M \rangle P'$ and $\{\mathbf{m}\} \cap (\text{fn}(P) \cup n(\pi_1(h))) = \emptyset$ then there exist $b \in \mathcal{N}$ and a concretion $(\nu \mathbf{n})\langle M \rangle Q'$ such that $h \vdash a \leftrightarrow b$, $\{\mathbf{n}\} \cap (\text{fn}(Q) \cup n(\pi_2(h))) = \emptyset$, $Q \xrightarrow{\bar{b}} (\nu \mathbf{n})\langle M \rangle Q'$ and $I(h \cup \{(M, N)\}) \vdash P'\mathcal{R}Q'$;
- if $P \xrightarrow{a} (x)P'$ then there exist $b \in \mathcal{N}$ and an abstraction $(y)Q'$ such that $h \vdash a \leftrightarrow b$, $Q \xrightarrow{b} (y)Q'$ and for all $B \subset \mathcal{N}$, where $|B| \leq 2^d$, $B \cap (\text{fn}(P) \cup \text{fn}(Q) \cup n(h)) = \emptyset$ and $h \cup id_B$ is consistent, for all pairs (M, N) of ground terms, if $\exists k \leq d$ $h \cup id_B \vdash_k M \leftrightarrow N$ then $h \cup id_B \vdash P'\{M/x\}\mathcal{R}Q'\{N/y\}$.

Definition 14 (d-hedged bisimulation and bisimilarity). A *d-hedged bisimulation* is a *d-hedged simulation* \mathcal{R} such that $\mathcal{R}^{-1} = \{(h^{-1}, Q, P) \mid h \vdash PRQ\}$, where $h^{-1} = \{(N, M) \mid (M, N) \in h\}$, is also a *d-hedged simulation*.

The *d-hedged bisimilarity*, written \sim^d , is the greatest *d-hedged bisimulation*, i.e. the union of all *d-hedged bisimulations*.

These definitions immediately lead to the following results.

- Proposition 1.** (a) Every hedged bisimulation is also a *d-hedged bisimulation*, for any $d \geq 0$.
(b) For any $d > 0$, a *d-hedged bisimulation* is also a $(d-1)$ -hedged bisimulation.

Proof. (a) By removing the cardinality constraint on the set B and requiring only its finiteness, we get the definition of hedged simulation; hence hedged simulations satisfy the definition of *d-hedged simulation* for any d .
(b) Let h be a hedge and $d > 0$; it is trivial to check that if $h \vdash P \sim^d Q$ then $h \vdash P \sim^{d-1} Q$. \square

We now aim to show that, for any two processes P, Q , there exists $d \geq 0$ such that $\exists h \in \mathcal{H} \ h \vdash P \sim^d Q \Rightarrow \exists h \in \mathcal{H} \ h \vdash P \sim Q$. This does not hold for arbitrary infinite processes since these can analyse messages of arbitrary depth. Therefore, we now consider only the fragment of spi-calculus without replication.

It should be noted that *let* and *guard* operators are the only constructs that can check the structure of messages. For instance, the process $c(x).[x = t].P$ uses

the term t to “test” a message received along channel c ; therefore, any message with depth greater than t ’s will fail the test, and hence we can consider to send the process only messages with depth up-to that of t . This observation leads to the following definition of *analysis depth*.

Definition 15 (Analysis depth). *Let P be a finite process. The analysis depth of P , denoted by $\text{ad}(P)$, is defined inductively by the clauses*

$$\begin{aligned} \text{ad}(0) &= 0 & \text{ad}(P|Q) &= \text{ad}(P) + \text{ad}(Q) \\ \text{ad}((\nu n)P) &= \text{ad}(P) & \text{ad}(\phi.P) &= \max(\text{ad}(P), \text{mcd}(\phi)) \\ \text{ad}(\overline{M}\langle N \rangle.P) &= \text{ad}(P) & \text{ad}(P + Q) &= \max(\text{ad}(P), \text{ad}(Q)) \\ \text{ad}(M(x).P) &= \text{ad}(P) & \text{ad}(\text{let } x = t \text{ in } P) &= \text{ad}(P\{t'/x\}) + \text{mdd}(t) \end{aligned}$$

where, in the case of the *let* operator, t' is any message such that $\text{mcd}(t') = \text{mcd}(t)$ (e.g., t' can be obtained by nesting $\text{mcd}(t)$ encryptions with a fresh name) and the maximal destructor depth of a term is defined as follows:

$$\begin{aligned} \text{mdd}(n) &= 0 & \text{mdd}(\{t_1\}_{t_2}) &= \max(\text{mdd}(t_1), \text{mdd}(t_2)) \\ \text{mdd}(x) &= 0 & \text{mdd}((t_1, t_2)) &= \max(\text{mdd}(t_1), \text{mdd}(t_2)) \\ \text{mdd}(\pi_i(t)) &= \text{mdd}(t) + 1 \quad i = 1, 2 & \text{mdd}(D_{t_2}(t_1)) &= \max(\text{mdd}(t_1), \text{mdd}(t_2)) + 1. \end{aligned}$$

Definition 16 (Critical depth). *Let P and Q be two finite processes and let h be a hedge. The critical depth $\text{CD}(h, P, Q)$ is defined by*

$$\text{CD}(h, P, Q) \triangleq \text{mcd}(h) + \max(\text{ad}(P), \text{ad}(Q)) + 1$$

Remark 1. In the definition of analysis depth (Definition 15) we have taken into account also the analysis done by matching operators. This differs from Hüttel’s work about framed bisimilarity, where in the definition of the analysis depth (there called “maximal destruction depth” [4, Def. 13]) it is $\text{ad}([M = N].P) = \text{ad}(P)$. In fact, it is crucial to consider also the matching operator. As an example, let us consider the following processes and frame-theory pair:

$$P = a(x).[x = \{a\}_a].\bar{a}\langle a \rangle.0 \quad Q = a(x).0 \quad (fr, th) = (\{a\}, \emptyset)$$

According to [4] the critical depth would be $\text{CD}((fr, th), P, Q) = 0$; hence $\{a\}_a$ would not be considered as a possible input message, since $\text{mcd}(\{a\}_a) = 1$ and $(fr, th) \not\vdash_0 \{a\}_a \leftrightarrow \{a\}_a$. Therefore P and Q would behave similarly for each input message tested by Hüttel’s algorithm, leading to incorrectly conclude that $(fr, th) \vdash P \sim Q$. This does not happen if the analysis depth takes into account the number of constructors used by matching operators, as in Definition 15. \square

As we will formally see below, when checking if a d -hedged simulation exists, we can correlate two input $P \xrightarrow{a} (x)P'$ and $Q \xrightarrow{b} (x)Q'$ w.r.t. a hedge h , simply by testing the equivalence between P' and Q' w.r.t. messages with *maximal constructor depth* less or equal to $\text{CD}(h, P, Q)$. Moreover, Lemma 1 limits the number of fresh names we must consider to $2^{\text{CD}(h, P, Q)}$.

We will now prove that, if P, Q are finite processes and $d = \text{CD}(h, P, Q)$ then

$$\exists h \in \mathcal{H} \ h \vdash P \sim^d Q \implies \exists h \in \mathcal{H} \ h \vdash P \sim Q.$$

This result is based on the definition of *d-pruning* and two lemmata that show the equivalence of the transition system when considering only messages M such that $\text{mcd}(M) \leq \text{CD}(h, P, Q)$.

Definition 17 (d-pruning). Let M and N be two messages and h a consistent hedge such that $h \vdash M \leftrightarrow N$. For $d \geq 0$, the *d-pruning* of M and N w.r.t. h , denoted by $\text{pr}_d(h, M, N)$ is defined by cases as follows:

- if $(M, N) \in h$ then for all d : $\text{pr}_d(h, M, N) \triangleq (h, M, N)$
- if $(M, N) \notin h$ then

$$\begin{aligned} \text{pr}_0(h, M, N) &= (h \cup \text{id}_{\{a\}}, a, a) \text{ where } a \in \mathcal{N} \text{ is fresh} \\ \text{pr}_{d+1}(h, \{U\}_J, \{V\}_K) &= (h', \{M'\}_J, \{N'\}_K) \\ &\quad \text{where } (J, K) \in h \\ &\quad \text{and } \text{pr}_d(h, U, V) = (h', M', N') \\ \text{pr}_{d+1}(h, (L_1, R_1), (L_2, R_2)) &= (h'', (L'_1, R'_1), (L'_2, R'_2)) \\ &\quad \text{where } \text{pr}_d(h, L_1, L_2) = (h', L'_1, L'_2) \\ &\quad \text{and } \text{pr}_d(h', R_1, R_2) = (h'', R'_1, R'_2). \end{aligned}$$

Intuitively, the *d-pruning* of a message pair (M, N) generates a message pair (M', N') where subterms appearing at levels greater than d are replaced by fresh names w.r.t. h , M and N .

Critical depth and *d-pruning* are readily extended to processes and messages:

$$\text{CD}(h, P) \triangleq \text{CD}(h, P, P) \qquad \text{pr}_d(h, M) \triangleq \text{pr}_d(h, M, M)$$

Lemma 2. Let $(x)P$ be an abstraction of a finite process, h be a hedge and $d = \text{CD}(h, P)$. For every message M ,

$$P\{M/x\} > P'_M\{M/x\} \iff P\{N/x\} > P'_N\{N/x\}$$

where the same reduction rule is used, $N = \pi_2(\text{pr}_d(h, M))$ and

- in the case of the guard reduction: $P'_M = P'_N$
- in the case of the let reduction: $P'_M = Q\{\llbracket t\{M/x\} \rrbracket / y\}$ and $P'_N = Q\{\llbracket t\{N/x\} \rrbracket / y\}$, for some Q .

Proof. Let us consider the case of *guard* reduction, i.e., $P\{M/x\} = (\phi.Q)\{M/x\}$ for some Q and $\llbracket \phi\{M/x\} \rrbracket = \mathbf{true}$. We only need to show that

$$\llbracket \phi\{M/x\} \rrbracket \iff \llbracket \phi\{N/x\} \rrbracket$$

We proceed by induction on the structure of ϕ . The inductive cases are easy:

$$\begin{aligned} \llbracket \neg\phi\{M/x\} \rrbracket &\iff \neg\llbracket \phi\{M/x\} \rrbracket \xLeftrightarrow{IH} \neg\llbracket \phi\{N/x\} \rrbracket \iff \llbracket \neg\phi\{N/x\} \rrbracket \\ \llbracket (\phi \wedge \psi)\{M/x\} \rrbracket &\iff \llbracket \phi\{M/x\} \rrbracket \wedge \llbracket \psi\{M/x\} \rrbracket \\ &\xLeftrightarrow{IH} \llbracket \phi\{N/x\} \rrbracket \wedge \llbracket \psi\{N/x\} \rrbracket \iff \llbracket (\phi \wedge \psi)\{N/x\} \rrbracket \end{aligned}$$

Let us consider the base case of matching predicate $[t_1 = t_2]$. If $\text{fv}(t_1) \cup \text{fv}(t_2) = \emptyset$ then the result of the matching is independent from the message and the lemma trivially holds. Otherwise, i.e. $\text{fv}(t_1) \cup \text{fv}(t_2) = \{x\}$, the matching $[t_1 = t_2]\{M/x\}$ is true if and only if $\llbracket t_1\{M/x\} \rrbracket \equiv \llbracket t_2\{M/x\} \rrbracket$. Moreover, the first condition of Definition 2 requires $t_1\{M/x\}$ and $t_2\{M/x\}$ to have the same type. Therefore, for each path in the syntactic tree of t_1 ending with an occurrence of x , there is an equivalent path in t_2 ending with either an occurrence of x or a message of the same type of M ; and vice versa for the occurrences of x in t_2 . Hence, without loss of generality we can limit ourselves to matchings of the form $[x = x]$ (which is trivial) and $[x = t]$ —which we show next.

If $\text{mcd}(M) \leq d$ then $\text{pr}_d(M) = M$ and the lemma is proved. If $\text{mcd}(M) > d$ then $\text{mcd}(M) \geq \text{mcd}(N) \geq d > \text{mcd}(t)$, since d is the depth of the pruning. Therefore it holds that $\llbracket [x = t]\{M/x\} \rrbracket = \llbracket [x = t]\{N/x\} \rrbracket = \mathbf{false}$ and the lemma is proved. Moreover, $P'_M = P'_N$.

Let us consider now the *let* reduction, i.e., $P\{M/x\} = (\text{let } y = t \text{ in } Q)\{M/x\}$ for some t, Q . If M is not analysed in t through decryption or projections (i.e., x does not occur inside decryption or projections in t), then the result of the reduction will happen regardless of M and therefore the thesis holds true. The same happens if $\text{mcd}(M) \leq d$, since $N = \text{pr}_d(M) = M$ and therefore the lemma holds again. Otherwise, if M is analysed in t and $\text{mcd}(M) > d$ then $\text{mcd}(M) \geq \text{mcd}(N) \geq d > \text{mcd}(t)$ and the reduction happens for $P\{M/x\}$ if and only if it happens for $P\{N/x\}$. From the definition of reduction for *let* reduction it follows that $P'_M = Q\{\llbracket t\{M/x\} \rrbracket / y\}$ and $P'_N = Q\{\llbracket t\{N/x\} \rrbracket / y\}$. \square

Lemma 3. *Let $(x)P$ be an abstraction of a finite process, h be a hedge and $d = \text{CD}(h, P)$. For every message M ,*

$$P\{M/x\} \xrightarrow{\alpha} P'_M\{M/x\} \iff P\{N/x\} \xrightarrow{\alpha} P'_N\{N/x\}$$

where the same reduction rule is used, $N = \pi_2(\text{pr}_d(h, M))$ and P'_M depends upon the application of the substitution $\{M/x\}$ and the transition rule used.

Proof. By induction on the transition rules. Let us start with the *parallel* rule. Let $P = Q|U$. By inductive hypothesis

$$Q\{M/x\} \xrightarrow{\alpha} Q'\{M/x\} \iff Q\{N/x\} \xrightarrow{\alpha} Q'\{N/x\}$$

and therefore, the application of the *parallel* rule leads to

$$(Q|U)\{M/x\} \xrightarrow{\alpha} (Q'|U)\{M/x\} \iff (Q|U)\{N/x\} \xrightarrow{\alpha} (Q'|U)\{N/x\}$$

and $P'_M = P'_N$. The same reasoning can be used for *sum* and *restriction* rules. For *equivalence* rules, only reduction rules modifies the processes substantially. Therefore, we can use Lemma 2 and the proof steps are equivalent to the ones of the *parallel* rule.

Let us consider now the *input* rule, i.e., $P\{M/x\} = (t(y).Q)\{M/x\}$. Since $(t(y).Q)\{M/x\}$ must be a ground process, $\text{fv}(t(y).Q) \subseteq \{x\}$ and t can only be

a name or equal to x . If $x \neq t \in \mathcal{N}$, then the transition is independent from M and will also occur for N . Otherwise, $t = x$ and the transition will occur only if M is a name. Moreover, if $M \in \mathcal{N}$ then $M = pr_d(M) = N$ and therefore the transition will also occur for $(t(y).Q)\{N/x\}$. It also holds that $P'_M = P'_N$.

The case for *output* rules is similar to *input*'s, since the transition of a term $(t(X).Q)\{M/x\}$ only depends on the channel t .

Let us now prove the lemma for *interaction* rules. By inductive hypothesis:

$$\begin{aligned} P\{M/x\} &\xrightarrow{n} ((y)P')\{M/x\} \iff P\{N/x\} \xrightarrow{n} ((y)P')\{N/x\} \\ Q\{M/x\} &\xrightarrow{\bar{n}} ((\nu m)\langle T \rangle Q')\{M/x\} \iff Q\{N/x\} \xrightarrow{\bar{n}} ((\nu m)\langle T \rangle Q')\{N/x\} \end{aligned}$$

therefore, the application of the *interaction* rule leads to

$$\begin{aligned} (P|Q)\{M/x\} &\xrightarrow{\tau} ((\nu m)(P\{T\{M/x\}/y\}|Q'))\{M/x\} \\ &\iff (P|Q)\{N/x\} \xrightarrow{\tau} ((\nu m)(P\{T\{N/x\}/y\}|Q'))\{N/x\} \end{aligned}$$

which is our thesis, and $P'_X = ((\nu m)(P\{T\{X/x\}/y\}|Q'))$ for $X \in \{M, N\}$. \square

Theorem 1. *Let P and Q be two finite processes. Then:*

$$\exists h \in \mathcal{H} \ h \vdash P \sim Q \text{ if and only if } \exists h \in \mathcal{H} \ h \vdash P \sim^{\text{CD}(h, P, Q)} Q$$

Proof. By Proposition 1, any hedged bisimulation is also a d -hedged bisimulation; hence, it suffices to show that if there is $h \in \mathcal{H}$ such that $h \vdash P \sim^{\text{CD}(h, P, Q)} Q$, then there is $h \in \mathcal{H}$ such that $h \vdash P \sim Q$. This follows from Lemmata 2 and 3, since we already shown that reductions and transition system is the same whenever we perform a pruning of depth $\text{CD}(h, P, Q)$. Moreover, it holds that

$$\begin{aligned} \mathcal{R} &= \{(h, P, Q) \mid \exists h' \in \mathcal{H} \ \exists P', Q' \in \mathcal{P} \ \exists M, N \in \mathcal{M} \text{ s.t.} \\ &\quad P = P'\{M/x\}, Q = Q'\{N/y\}, (h', M', N') = pr_d(h, M, N), \\ &\quad h' \vdash P'\{M'/x\} \sim^d Q'\{N'/y\}, \text{ for } d = \text{CD}(h, P, Q)\} \end{aligned}$$

is a hedged bisimulation, since for a transition α it holds that

$$\begin{aligned} P'\{M/x\} &\xrightarrow{\alpha} P''\{M/x\} \xleftarrow{\text{Lemma 3}} P'\{M'/x\} \xrightarrow{\alpha} P''\{M'/x\} \\ &\xrightarrow{\sim^d} Q'\{N'/y\} \xrightarrow{\alpha} Q''\{N'/y\} \xleftarrow{\text{Lemma 3}} Q'\{N/y\} \xrightarrow{\alpha} Q''\{N/y\} \end{aligned}$$

and vice versa. Furthermore there exists h'' , obtained by updating h with the effects of α , such that, from Definition 13:

$$(h''', M'', N'') = pr_{d'}(h'', M', N') \text{ and } h''' \vdash P''\{M''/x\} \sim^{d'} Q''\{N''/y\}$$

where $d' = \text{CD}(h'', P'', Q'')$. Therefore $(h'', P''\{M/x\}, Q''\{N/y\}) \in \mathcal{R}$. \square

Since every quantification is bounded, d -hedged bisimilarity is decidable on finite processes. An algorithm is shown in Figure 3. For P, Q two finite processes and h a hedge (which represents the initial knowledge of the attacker, e.g. public channels, keys, etc.), $\mathcal{HB}(h, P, Q) \wedge \mathcal{HB}(h^{-1}, Q, P) = \mathbf{true}$ if and only if there is a d such that (h, P, Q) are in a d -hedged simulation. Hence, by Theorem 1:

$$h \vdash P \sim Q \iff \mathcal{HB}(h, P, Q) \wedge \mathcal{HB}(h^{-1}, Q, P).$$

```

1  $\mathcal{HB}(h, P, Q) =$ 
2   for each  $P \xrightarrow{\tau} P'$ 
3     select  $Q \Rightarrow Q'$  such that
4        $\mathcal{HB}(h, P', Q') \wedge \mathcal{HB}(h^{-1}, Q', P')$ 
5   for each  $P \xrightarrow{\bar{a}} (\nu m)(t)P'$ 
6     select  $Q \xRightarrow{\bar{b}} (\nu n)(t')Q'$  such that
7        $h_O := I(h \cup \{(a, b)\} \cup \{(\llbracket t \rrbracket, \llbracket t' \rrbracket)\})$  consistent and
8        $\mathcal{HB}(h_O, P', Q') \wedge \mathcal{HB}(h_O^{-1}, Q', P')$ 
9   for each  $P \xrightarrow{a} (x)P'$ 
10    let  $d = \text{CD}(h, P, Q)$ 
11    select  $Q \xRightarrow{b} (x)Q'$  and  $B \subset \mathcal{N}$  such that
12       $|B| = 2^d, \quad B \cap (\text{fn}(P) \cup \text{fn}(Q) \cup n(h) \cup \{a, b\}) = \emptyset,$ 
13       $h_I := h \cup \{(a, b)\} \cup \text{id}_B$  consistent and
14      for each  $(M, N)$  such that  $\exists k \leq d: h_I \vdash_k M \leftrightarrow N$ 
15         $\mathcal{HB}(h_I, P'\{M/x\}, Q'\{N/x\}) \wedge \mathcal{HB}(h_I^{-1}, Q'\{N/x\}, P'\{M/x\})$ 

```

Fig. 3. Algorithm for deciding hedged bisimilarity. The *select* statement implements a nondeterministic exploration of the (finite) possible choices of its argument, until the condition is satisfied; it returns **true** if successful, **false** otherwise.

5 Conclusions and further work

In this paper we have proved that hedged bisimilarity is decidable on finite processes of the spi calculus. Our algorithm, which generalizes the ideas in [4], can be readily applied to different encryption/decryption schemata just by changing the congruence rules, as long as some mild conditions are satisfied. Actually, a possible future work is to investigate the algebraic laws needed to represent in the structural congruence the properties of various encryption algorithms: often these laws are omitted from formalizations, leading to security flaws in protocols. Another direction is to consider other fragments of the spi-calculus beyond finite processes; depth- and restriction-bounded processes are particularly promising.

References

1. Martín Abadi and Andrew D. Gordon. A bisimulation method for cryptographic protocols. *Nordic Journal of Computing*, 5(4):267–303, 1998.
2. Martín Abadi and Andrew D. Gordon. A calculus for cryptographic protocols: The spi calculus. *Inf. Comput.*, 148(1):1–70, 1999.
3. Johannes Borgström and Uwe Nestmann. On bisimulations for the spi calculus. *Mathematical Structures in Computer Science*, 15(3):487–552, 2005.
4. Hans Hüttel. Deciding framed bisimilarity. In *Proceedings of Infinity’02*, volume 68 of *Electronic Notes in Theoretical Computer Science*, pages 1–18, 2003.